End-User Centered Events Detection and Management in the Internet of Things

Stefano Valtolina(✉), Barbara Rita Barricelli, and Marco Mesiti

Department of Computer Science, Università degli Studi di Milano, Milano, Italy
{valtolin,barricelli,mesiti}@di.unimi.it

Abstract. With the widespread of Internet of Things' devices and sensors, the use of applications for combining data streams is increasing in a high number of dynamic contexts, with a huge variety of users having radically different backgrounds and a plethora of diverse tasks to perform. On the one hand, users need a system able to support them in detecting significant events with respect to their context of use. On the other hand, such users need an environment in which they can express proper requirements for enabling the system to react autonomously according to events in the real/physical world, by running processes that trigger actions and perform services. In this paper, we provide the design of a web environment that we are developing around the concept of event, that is simple or complex data streams gathered from physical and social sensors that are encapsulated with contextual information (space, time, thematic). Moreover, from a study of the most diffused applications for Internet of Things that provide the user with tools for controlling the combination of data streams coming from devices and sensors, we identify and discuss some open problems and proposes a new paradigm and language to address them and to be integrated in our web environment.

Keywords: Internet of things · Event detection · Pervasive computing · Unwitting developers · End users

1 Introduction

Nowadays we are witnesses of the proliferations of different sensor devices able to produce heterogeneous types of data (textual, visual, audio, and other rich multimedia formats) that can be profitable used for detecting, handling and advising people of the verification of particular conditions about physical phenomena (like temperature or humidity). These events are characterized according to different dimensions such as: the time (when they happen), the space (where they happen) and the thematic (what they concern) that can be exploited from one side for the identification of the useful information needed to face a given event and from another side for the analysis and forecast of useful activities to carry out. Conventional data analytics platforms cannot be exploited profitably for handling this kind of data and new advanced architectures should be developed for several reasons. In this context, network configuration, sensor detection and discovery are difficult issues to be solved. Moreover, sensors and

F. Daniel and O. Diaz (Eds.): ICWE 2015 Workshops, LNCS 9396, pp. 77–90, 2015.
DOI: 10.1007/978-3-319-24800-4_7

the data they produce should be handled in real time in order to be properly elaborated during the event. Therefore, scalable and efficient solutions should be devised that can be applied on-line. ETL (extract, transform and load) solutions, usually applied off-line, need to be revised and applied on-line for generating fresh and timely data. Finally, user-friendly environments have to be conceived for properly helping the users in combining the data coming from different sensors and devices and for enabling them to take under control the huge amount of data and events that can be generated. In this paper, we propose a novel event-based data model for manipulating multi-dimensional relations between multiple data streams based on event dimensions (spatial, temporal and thematic dimensions). Then, we propose a new language and paradigm for End-User Development that can be exploited in the context of Internet of Things (IoT). Specifically, we propose a sensor-based rule language able to support end users in aggregating and combining data originated by several sensors/devices. This language aims at enabling end user for unwittingly developing personalized IoT environments according to specific temporal, spatial, and fuzzy conditions that may affect the elements in the IoT environment.

The paper is organized as follows. In Section 2, the new architecture will be illustrated and compared with respect to conventional Data Warehouse (DW) systems. Section 3 describes the event data model tailored for handling multidimensional complex events according to a multigranular spatio-temporal-thematic (STT) approach. Section 4 presents the current state of the art of End-User Development (EUD) strategies in IoT and the applications that enable the users to arrange data coming from IoT devices/sensors. Section 5 starts with the ETL operations that are at the base of our event management model, and continues with the user interface developed for the application of such operations in a workflow based on the combinations of data coming from different sensors and devices. At the end of the section, the current state of the art of End-User Development (EUD) strategies in IoT is discussed along with the applications enabling the users to arrange data coming from IoT devices/sensors. Finally, Section 6 introduces a new EUD paradigm and language in IoT domain.

2 Research Background: From Conventional to Event based Analytic Platform

With OLAP ("Online Analytical Processing") systems, users can analyze data through the application of different cube operators applied to the data-warehouse (DW). For example, a company might want to compare their sales in June with sales in July, and then compare those results with the sales made in another location, which might be stored in a different database. Therefore, a data warehouse can be viewed as a large repository of historical data organized in a pertinent way in order to perform analysis and to extract interesting information using OLAP technology. Nevertheless, traditional data warehouse and OLAP systems are not able to support users in analyzing highly dynamic data processing and to generate meaningful analytical reports. This is particularly true in IoT systems where the data-streams are characterized by the 3V dimensions Volume, Variety, Velocity [1], along with the issue of integrating information coming from heterogeneous networks. In order to endow DWs and OLAP with these capabilities, recently the term "active" or "real-time" DW [2] was coined

to capture the need for a DW containing data as fresh as possible. In traditional DWs, there is a distinction between ETL (Extract, Transform, and Load) and OLAP operations. The former are used for feeding the DW, whereas the latter are used to query the DW once the cube has been defined. Moreover, feeding the DW is considered a pre-processing phase that is taken offline with the purpose to solve many issues (extracting, cleaning and riconciliate heterogeneous data, and denormalize data) in order to make somehow simpler the execution of the OLAP operations. Stream DW have been also proposed for handling information produced in streams and approaches based on sliding windows are used to manage continuous data [3].

In IoT the periodic population of a DW is considered outdated and new ETL and OLAP services are required in order to work in real time (or almost real time), that is, data are loaded in real time from the OLTP (Online Transaction Processing) systems into the DW, providing a convenient way for user to real-timely read the data information and make tactical decisions. In this context, the ETL and OLAP operations must be tightly connected for making the process possible and their implementations need to be quite efficient and scalable.

In our work, we wish to develop real-time DW that is developed around the ECA (Event, Condition, and Action) paradigm [4] because it is the most promising approach for the development of the behavior of an intelligent environment. We identify an event with both its temporal, spatial and thematic dimensions that can be exploited from one side for the identification of the useful information needed to face a given event and from the other side for the analysis and forecast of useful activities for notifying people. First, we need to point out the adoption of a different data model. Indeed, conventional DWs adopt the relational model for representing stored information, but this is not adequate in Event-based warehouse because of the heterogeneity of the data formats and also the fact that information is stored at different granularities (ranging from simple raw values to structured and complex data). Second, conventional systems process stored data, whereas in the Event-based warehouse also streams of data should be processed, collected and analyzed through temporal windows. Contextual information can be associated with the stream by means of machine learning algorithms and exploited for extracting knowledge. For what concerns the implementation of the architecture and of the required operations, we are considering the use of cloud-based architectures because we need to guarantee high throughput and low end-to-end latency from the system, despite possible fluctuations in the workload. We are currently evaluating several architectures like Apache S4 [5], D-Streams [6], Storm [7] and StreamCloud [8].

3 STT Event Data Model

In order to represent events, we consider a set of basic and structured types (like list and records) whose values are represented thorough a JSON-like notation. These types allow the representation of different aggregation of values without imposing the restrictions of the relational model and so are able to handle a great variety of datatypes. Moreover, we consider spatial and temporal types at different granularities. Temporal granularities include seconds, minutes, days with the usual meaning

adopted in the Gregorian calendar, whereas, meters, kilometers, feet, yards, provinces and countries are examples of spatial granularities. Different granularities provide different partitions of their domains because of the diverse relationships that can exist among granularities, depending on the inclusion and the overlapping of granules [9].

For example, in temporal granularity seconds *is finer than* minutes, and granularity months *is finer than* years. Likewise, spatial granularity provinces *is finer than* regions. Thematics are also considered for associating a semantic annotation to a given value. For example, the annotation "very high" can be associated to the event representing the degree (high) of temperature that is detected in a given area. Several thematics can be identified depending on the context where the datum is acquired or processed. Thematic can indeed be inferred by machine learning algorithms. Relying on the concepts of temporal and spatial granularities, we present the concept of event, that is a value associated with a spatial object at a given time according to given thematics. Therefore, an event is a value represented at a given spatio-temporal granularity for which thematic information is added. Relying on the concept of event, we can characterize an event stream that a source can produce. A source can be sensor (either physical or virtual) or a service (for example by aggregating sensors/services streams). As an example, let us consider a gym scenario where a trainer wishes to monitor the physical conditions of her/his users in order to suggest better exercises. These exercises need to be compliant to user's biological parameters such as the heartbeat variability or the weight and they need to be used on a fitness device such as a treadmill according to a plan of execution (the time that the user has to run, speed and inclination of the treadmill). The simple events, collected by these types of sensors and devices, can be characterized by the time dimension (when the biological parameters have been detected), the space dimension (where the biological parameters have been detected: at home, into the gym, or where the treadmill is located). The thematic of a simple event is the meaning associated to a value or a set of values. For example, the event: "Tachycardia" can be related to a very high number of heartbeats. Instead, the event: "Hard run" can be related to high inclination of the treadmill that works fast. The outcome the trainer expects is a set of complex events that are the actual exercises to provide to the users according to their physical conditions. In addition, these events have time dimension (when the exercise has to be executed) and space dimension (where the exercise has to execute in case the gym has more than one treadmill of different types). Moreover they have a thematic, which in this case is a complex document representing the execution plan of the exercise and the related user's physical condition (as a combination of heartbeat variability and weight).

4 End-User Development and Internet of Things

Shneiderman defined in [10] his concept of "old computing", focused on what computers can do for the user. On the other hand, today we deal with the "new computing" concept that refers to people activity and what people can actively do by using computers. Users of digital devices and interactive systems are increasingly evolving from passive consumers of data and computer tools into active producers of

information and software [11][12]. The potentials offered by network and connectibility of the objects do not only enrich the person's personal sphere but also give the chance of sharing data with other people like family members, friends, colleagues, or others. In this scenario, end users find themselves at the center of a complex scenario that they need to manage in efficient, effective, and satisfactory way. EUD represents the ideal approach for empowering end users and make them becoming unwitting developers in their own IoT environment [13][14][15]. As widely reported in the literature, EUD can be enabled by applying methods and techniques and by offering specific tools that allow end users to develop solutions without having specific programming skills and knowledge about programming languages. Specifically, the solutions offered by EUD include tools for the customization of applications by parameters setting, control of a complex device (like a home-based heating system), and even scripting of interactive Web sites [16]. EUD allows users to configure, adapt, and evolve their software by themselves [17] and such tailoring activities, together with personalization, extension, and customization are defined in literature in different ways, sometimes referring the same concepts and sometimes referring different ones [18]. The target of EUD activities in the IoT context is not (only) the user interface and the behavior of an interactive system, but embraces the whole IoT scenario of sensors and devices. Therefore, we need to distinguish between those activities that can be made at three different levels: hardware, software, and data. EUD activities on hardware are those made on the devices via their bundled applications. They typically are configuration, personalization, and customization by setting parameters and choosing among existing behaviors. The activities on software target concern the applications used for controlling more than one sensor/device (even of different brands) and include tailoring by integration of existing and/or new functionalities, macros, visual programming, and programming by examples. The EUD activities that can be made on data can be resumed in aggregation, filtering, and porting. In what follows, we will use the classifications presented so far for discussing the state of the art of applications that can manage data originated by more than one sensor/device and shared on Social Media, and that enable end users to unwittingly develop their own IoT environment. In Section 6, we provide an overview of the current state of the art in IoT applications/tools that offer EUD activities to their users and we relate them with the new paradigm that we propose in this paper.

5 Event Management

In our STT data model, one of the most important EUD activity takes place during the design of the most relevant and significant events that characterize a context of use. For example in a gym, the trainers need to combine data coming from different sensors and devices for properly defining the exercises. These EUD activities are exploited for executing efficiently and effectively Event ETL operators over a programmable network (e.g., filter by in-network data processing). So far, we have developed three kinds of operators that are relevant in our context: conversion, merging, and connecting operators. Conversion operators are used for changing the

spatio-temporal-thematics granularity of an event stream. Merging operators are used for making the union of events at the same spatio-temporal granularity in a single data structure. Connecting operators are used to link a data source to another one by applying a SQL joint-like operator. In the rest of the section, we focus on the graphical visual interface using which the domain expert, not necessarily expert in IoT technologies, can identify the sensors and apply the operators for converting, merging and combining their data stream.

5.1 Visual Event ETL

ETL operations have been proposed in different contexts depending on the kinds of data to handle (structured and semi-structured). In [2][3][19] there is a good treatment at the conceptual level for feeding a DW. Moreover, in [20] there are approaches for the semi-automatic generation of ETL operations depending on the user needs and context of use. ETL operations are usually coupled with graphical visual data-flow for helping the user in the identification of the original data sources, the application of the operations for extracting, cleaning, transforming and combining their data. In our case, the user is a domain expert that needs a visual interface for combining sensors, and services that are used for generating a flow of significant events to be provided to the final user. For example in a gym, a trainer possibly needs to integrate fitness devices with personal data of her/his users and with other data streams coming from personal sensors used for capturing biological parameters. Once the ETL specification is completed, some strategies are proposed for the optimization of the dataflow and for the efficient execution of the loading schedule. In the fitness case study, the events might concern the user's physical condition, which will be used for suggesting what machine is better to use or which exercises. These approaches have been mainly developed for producing relational data to feed conventional DW system. In [21] an approach is presented for feeding arbitrary target sources (either relational or based on a NoSQL system). In our context, we designed a Web environment where the domain expert can drag and drop different sensor data sources and visually apply on them a set of operations. This application offers an engine and graphical environment for data transformation and mashup. As depicted in Fig. 1 (section A), this mash-up consists of a user interface that contextually displays icons of data sources or operations in order to link, filter or merge data coming from different sensors. It relies on the idea of providing a visual workflow generator for letting the domain experts to create aggregation, filtering, and porting of data originated by sources. An advanced use of such visual paradigm allows the domain experts to have an online generation of sample data coming from the data sources dragged-and-dropped on the canvas, or as result of the operations carried out on them (see Fig. 1 section B). By this strategy, information is gathered from the net and specific filters and operations can be triggered on user requests. This is a very simple and easy to learn solution based on the definition of sources to use for collecting data and on the possibility to apply our operations in a visual way. Future activities aim at developing machine-learning algorithms for exploiting the possible operations that domain experts can apply to a set of data

sources or for providing them useful predictions of what can happen by integrating the selected data.

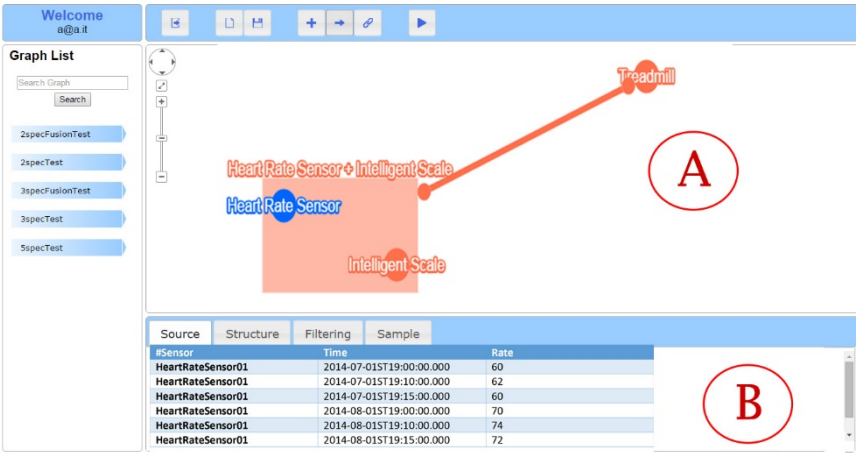

Fig. 1. A screenshot of the user interface of the Visual Event ETL

6 End-User Empowerment for Controlling Events

Currently several tools in IoT context are applying the ECA paradigm for supporting users in generating personalized actions according to a set of self-defined conditions. Users can monitor through them specific events of their environment, and some examples are IFTTT (https://ifttt.com) and Atooma (http://www.atooma.com). By analyzing these services, we identified two main types of applications that differ in terms of activities and interaction style. The first one allows users to define sets of desired behaviors in response to specific events. This is made mainly through rules definition-wizards that rely on the sensors/devices states. Rules can be typically chosen among existing ones or can be tweaked through customization. These EUD activities put in place a task automation layer across all sensors/devices in the IoT environment. Such strategy is adopted by those applications that use automated rules-based engines like Atooma and IFTTT– by using the programming statement IF this DO that, and by Wewiredweb (https://wewiredweb.com/) with the statement WHEN trigger THEN action. A more advanced use of these applications sees the involvement of recommendation systems (RSs) as part of the IoT ecosystem. In these cases, EUD activities are supported by the RSs that, relying on end user's pattern of use of devices, can suggest compelling examples of statements that the user can adapt to his needs.

A second type of applications stems from the outstanding work done with Yahoo's Pipes (https://pipes.yahoo.com/pipes/). These solutions typically use formula languages and/or visual programming. Applications like Bipio (https://bip.io/) and DERI pipes (http://pipes.deri.org/) offer engine and graphical environment for data transformation and mashup. They are based on the idea of providing a visual pipeline

generator for supporting the end user in creating aggregation, filtering, and porting of data originated by sources. An advanced use of such visual paradigm is offered by WebHooks (https://developer.github.com/webhooks/) that allows the end users to even write their personal API for enabling connections with new sources of data. Both propose EUD strategies, adoptable in the context of the IoT applications, for gathering information from across the net and triggering specific actions when certain things happen. The first type of applications offers a very simple and easy to learn solution based on the definition of ad hoc rules that can notify the end users when something happens – e.g. when their favorite sites are updated, when they check-in in some places or their friends do, or warn them when specific weather conditions are going to take place. However, the adoption of the IF-THIS-THAN-THAT/WHEN-TRIGGER-THEN-ACTION patterns is not enough to deal with more sophisticated rules based on time, space, and fuzzy conditions. On the other hand, the second type of applications offers a too complex solution for supporting the end user in expressing their preferences. Pretending that end users are able to deal with APIs of several sensors/devices put at risk the success of the EUD approach. Another issue of the current proposals regards the use of the social dimension in the most diffused applications, while time and space dimensions are almost neglected.

To face these problems, in the next section we propose an extension of the IF-THIS-THEN-THAT paradigm by presenting a sensor-based rule language able to support the end user in defining rules in a more articulated way but keeping the complexity at an acceptable and accessible level. This idea is to keep the simplicity of the IF-THIS-THEN-THAT paradigm pairing it with the use of formula languages. Moreover, time and space dimension will be exploited and fuzzy conditions adopted for expressing more loose rules in the statements.

6.1 A New EUD Paradigm and Language for IoT

In order to manage the great flow of events generated by the event-based DW systems and the flow of data-streams provided by sensors it is necessary to devise an innovative strategy for supporting the end user empowerment. To make the users feel like they are in control, an ECA-based rule strategy can be a promising solution. Through the definition of a set of conditions, users can enable the system to suggest some suitable actions according to their needs and context of use. The flow control determines how the system will respond when certain conditions are in place and specific parameters are set. By apply this strategy, the end user needs to state a if condition (e.g. the weather station says that it is going to rain in the next 12 hours) and an action (e.g. tweet this news on the end user's Twitter personal account using hashtags #weather #rain). As described before, this solution is adopted in many applications for supporting end users in creating rules for their IoT environment. The IF-THIS-THEN-THAT paradigm seems to work well when end users need to be warned or notified on a specific event, but uses a very simple language that has a quite low expressive power. By exploiting the three dimensions defined for our STT Event Data Model, we propose a new rule-base language able to give to end users the possibility of setting triggers that do not depend just on one simple event but also on complex events. The introduction

of the time dimension allows end users to set triggers that can be fired at some specific time, delayed in case of certain conditions are verified, and may be repeated until some event happens. The space dimension gives end users the chance of linking triggers to the place/area where they currently are, where they will possibly be in the future, where they are moving into, or where some events are taking place. In [4][22] there is a nearly unanimous agreement on an extension of "classical" trigger languages by including time dimension.

The proposal in this field can be summarized as follows: (1) Rules should be triggered by the occurrence of the dimension of events, (2) Enabling periodically repeated triggering of the same reaction, where the period is specified by an expression returning a time duration. (3) Delaying reaction execution to some later point in time relative to the triggering event of a rule. In [23] the authors propose a set of functionalities to be implemented with triggers written in SQL:1999 standard that cover three types of temporal categories – absolute, periodic, and relative event specifications – and allow to base delay or periodic repetition on valid time or transaction time events, respectively. According to this proposal of functionalities, we can provide users with a new set of rules composition-strategies able to go beyond the simple use of an IF-THIS-THEN-THAT statement. Up to now, rules can be triggered by call events only, and reactions are always executed one time. We identified four types of rule:

1. Space Events: rules that need to be triggered if the data stream refers to a specific geographical place/area. An example: IN *"home place" IF "I do not enter in the gym-room at least three times in a week, send me an alert"*.
2. Time Events: Rules triggered on certain absolute time events are the most common feature of time-triggers. An example: AT *"summer time",* IF *"my sleep-controller detects that I did not sleep well last night"* THEN *"my activity tracker device should suggest me to take a walk before going to bed"*.
3. Delayed Reaction Execution: Reaction execution can be delayed by combining a call event with a temporal offset. This offset is a time-valued attribute of the related environment, thus generating "relative events". For instance, to check three months after my last blood test if I need another test, a possible rule could be: AT *"The date of my last blood test + 3 months"* IF *"the person scale says that I lost more than 10 kilograms"* THEN *"my smart watch should show a message suggesting to book a medical exam"*.
4. Repeated Reaction Execution: Repeating execution of a particular action regularly after a fixed period has passed. In this case the keyword EVERY could be combined with an expression of type PERIOD, e.g. EVERY MONTH, or EVERY 3 HOURS, or with even more sophisticated specifications, such as EVERY MONDAY, EVERY 2nd MONDAY IN A MONTH, or EVERYDAY EXCEPT SATURDAY.

The EUD paradigm we propose in this paper aims supporting the end user in composing such space/time-based rules for extending the well-established but not powerful IF-THIS-THEN-THAT paradigm. Our Sensor-based Rule Language follows syntax and semantics of a Policy Rule Language was proposed in [25], and is based on the ECA paradigm. Our language allows the specification of rules stating policies for

triggering actions (one or a set). The general format of a rule is the following (square brackets denote optional components):

```
RuleName: "MY RULE"
ON SOURCE[s]
  [WHENEVER "Condition"]
Action: "Some Actions"
  [VALIDITY: Validity_Place-Interval]
```

A rule consists of several components. The `RuleName` component represents the rule identifier. Users can retrieve rules by means of such identifier for visualizing, sharing, dropping, or modifying them. `SOURCE[s]` represents the source or set of source upon which data the rule is triggered. In our case the user can take data from a sensor for example a thermometer, a service for example the weather forecast service or an event-based DW system. The last source provides users with the possibility to take into account complex events generated by domain experts in specific context of use as for example in the fitness case study previously described or in other contexts such as traffic control or healthcare or emergency situations.

Each source exposes a set of data or events, which can be used for expressing the conditions. `Condition` is an optional conditional expression. `Action` is an expression that states what happens when the condition is verified. `Validity_Place-Interval` is a special spatial and/or temporal condition also expressed by means of the condition language we developed, representing the space and time period during which the rule is enabled. For example, if the interval [EVERYDAY EXCEPT SATURDAY] is specified we know that a rule is enabled every day of the week but not on Saturday. However, if `Validity_Place-Interval` is not specified, we know that the rule is always enabled. By means of `Validity_Place-Interval` it is possible to state that certain rules are not always enabled; rather, they are enabled only if an event happens in a specific place or during specific temporal intervals. Such a feature is not provided by conventional apps for IoT. A possible rule generated in the fitness case study might be:

```
RuleName: "Next week exercises"
ON GYM Event-Base DW AND Thermometer AND Weather forecast
service
  WHENEVER "the suggested exercises concern the use of a
treadmill" AND "the temperature is not more than 30 de-
gree" AND "the weather is forecasted good"
Action: "suggest me to run outside into the park"
  VALIDITY: IN "Milan" and AT "June-August"
```

In case the user involves event-based DW systems in the definition of the rule, she/he has the possibility to trigger more complex actions according to the parameters of the events provided by the DW. For example if the trainer describes the exercise with an indication of the rate of difficulty the user can create a new rule for asking for a simpler exercise in case the sleep controller detects that last night she/he slept few hours.

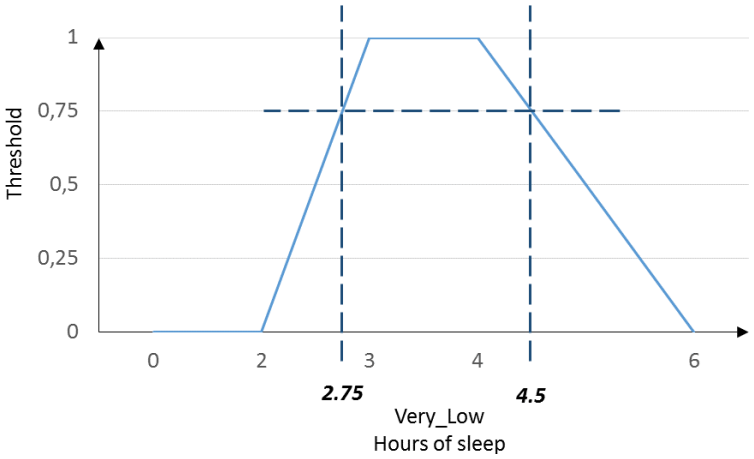

Fig. 2. Example to illustrate hours of sleep distribution.

Another extension of the IF-THIS-THEN-THAT paradigm is to provide end users with a more flexible way for expressing the condition statement by incorporating fuzziness into the condition on which the events need to be triggered. This extension focuses on the linguistic values of the fuzzy condition [25], in which the fuzziness concerning the linguistic concepts is interpreted in an application context. A linguistic variable is represented by a quintuple of <v,T,X,g,m> where v is name of the linguistic variable, T is set of linguistic terms applicable to variable v, X is the universal set of values, g is the grammar for generating the linguistic term, m is the semantic rule that assigns to each term tϵT, a fuzzy set on X. To illustrate our approach, we use as example a sleep monitor. Let represent a linguistic variable with a graphical distribution based on four parameters as depicted in Figure 3. Very_Low for hours of sleep is represented using trapezoidal function as Very_Low (2 hours, 3 hours, 4 hours, and 6 hours). Current IoT applications use simple statements such as "Hours of sleep <= 3 hours" to indicate when the value is very low. Using our language, users can use the statement "Quantity = $Very_Low": a set of values are related to "$Very_Low" in this comparison, rather than one single value. Fulfillment threshold is allowed to specify the condition with a degree value in the range of [0, 1]. For example, in Figure 3, we used 0.75 as the threshold to indicate that the value of hours of sleep is very low with the degree of 0.75. As a result, a value in the range [2.75, 4.5] indicates that "the number of hours of sleep is very high with a threshold of 0.75". By using our Rule Language the user can express a fuzzy condition is this way:

```
RuleName: "Quality of sleep Monitor"
ON sleep-controller AND Thermometer
  WHENEVER "the number of hours of sleep is $Very_Low"
AND "the temperature is not $Very_High"
Action: "The activity tracking device suggests me to take
a walk before going to sleep"
```

7 Conclusion and Future Steps

In the paper, we presented the architecture of a web environment for gathering, computing, and diffusing data originated and streamed by physical and social sensors. The architecture is based on event data model designed for an approach that focuses on space, time and themes. After presenting strategies of End-User Development in the IoT context, aimed at giving the end users more freedom and power to assemble different data sources/sensors in an ad-hoc and personalized solution, we proposed a EUD paradigm and related language that is currently under development in an IoT application for wellness domain. In order to manage the overload of information coming from different sensors and event-based DW systems and to mitigate the overflow of activities that the user has to perform for taking all under control, future studies aim at investigating a solution for predicting ECA rules according to the user's behavior and practices. In a unware way the user will have the possibility to experiment new rules predicted by an intelligent system by applying a serendipity-based strategy. The suggestion can be formulated by taking into account the user's profile and recurring activities carried out by using the combination of sensors and event-based DW systems. Moreover, a rating process of the suggestions provided to the user will enable an effective human control over the intelligent algorithm for improving the set of recommendations. This because by relying only on automatic suggestions may not be entirely appreciated by the user, which risks to get frustrated in using the recommendation system whenever the recommendations prove to be wrong for her/him. To solve this problem, the main challenges that we have to face are: (i) The need of a recommendation model and, as a consequence, of a recommendation policy understandable by the user; (ii) The need to define how the user can be part of the recommendation process; (iii) The need to exploit the social relationships between users. User-centered recommendations can be enriched in considering social networks where each user will certainly appreciate receiving recommendations from those considered "closest" to her/him; (iv) The need to take into account the uncertainty of both the ratings/suggestions proposed by the social networks members. In fact, judgments collected from a plethora of users with different habits and cultures may produce contradictions, which in turn result in rules having a high degree of ambiguity. This calls for a granular interpretation of the information provided by such crisp attributes, for instance in terms of fuzzy sets, rough and interval sets, and so on.

In conclusion, the definition of an innovative system able to balance the human and the machine empowerment by adopting a programming by practices paradigm will allow us the unprecedented possibility to assist users or groups of users in completing their daily tasks in an unaware way.

References

1. Schroeck, M., Shockley, R., Smart, J., Romero-Morales, D., Tufano, P.: Analytics: The Real-World Use of Big Data (IBM Institute for Business Value - Executive Report). IBM Institute for Business Value (2012)
2. Gorawski, M., Gorawska, A.: Research on the stream ETL process. In: Kozielski, S., Mrozek, D., Kasprowski, P., Małysiak-Mrozek, B. (eds.) BDAS 2014. CCIS, vol. 424, pp. 61–71. Springer, Heidelberg (2014)
3. Imran, M., Castillo, C., Diaz, F., Vieweg, S.: Processing Social Media Messages in Mass Emergency: A Survey eprint arXiv:1407.7071 (2014)
4. Widom, J., Ceri, S.: Active Database Systems. Morgan Kaufmann Publisher (1996)
5. Zhou, H., Yang, D., Xu, Y.: An ETL strategy for real-time data warehouse. In: Wang, Y., Li, T. (eds.) Practical Applications of Intelligent Systems. AISC, vol. 124, pp. 329–336. Springer, Heidelberg (2011)
6. Neumeyer, L., Robbins, B., Nair, A., Kesari, A.: S4: distributed stream computing platform. In: IEEE Int'l Conf. on Data Mining Workshops, pp. 170–177 (2010)
7. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., Stoica, I.: Discretized streams: fault-tolerant streaming computation at scale. In: ACM Symposium on Operating Systems Principles, pp. 423–438 (2013)
8. Marz, N.: Storm: Distributed and fault-tolerant realtime computation (2012)
9. Gulisano, V., Jiménez-Peris, R., Patino-Martinez, M., Soriente, C., Valduriez, P.: Streamcloud: An elastic and scalable data streaming system. IEEE Trans. Parallel Distrib. Syst. **23**(12), 2351–2365 (2012)
10. Shneiderman, B.: Leonardo's Laptop: Human Needs and the New Computing Technologies. MIT Press (2002)
11. Fischer, G.: Beyond 'Couch Potatoes': From Consumers to Designers and Active Contributors" (2002). http://firstmonday.org/issues/issue7_12/fischer/ (accessed on January 9, 2015)
12. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: a Model-based Design Methodology. IEEE TSMCA **37**(6), 1029–1046 (2007)
13. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: End users as unwitting software developers. In: Proc. of WEUSE 2008, pp. 6–10. ACM (2008)
14. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Advanced Visual Systems Supporting Unwitting EUD. In: Proc. of AVI 2008, pp. 313–316. ACM (2008)
15. Barricelli, B.R., Marcante, A., Mussio, P., Parasiliti Provenza, L., Valtolina, S., Fresta. G.: BANCO: a Web Architecture Supporting Unwitting End-User Development. IxD&A, 5-6, pp. 23–30 (2009)
16. Sutcliffe, A., Mehandjiev, N.: Introduction of Special Issue on End-User Development. CACM 47(9), 31–32 (2004)
17. Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V.: Introduction. In: Proc. of IS-EUD 2009, pp. V–VI. Springer (2009)
18. Mørch, A.: Three levels of end-user tailoring: customization, integration, and extension. In: Computers and Design in Context, pp. 51–76. MIT Press (1997)
19. Camossi, E., Bertino, E., Mesiti, M., Guerrini, G.: Handling expiration of multigranular temporal objects. J. Log. Comput. **14**(1), 23–50 (2004)
20. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual modeling for ETL processes. In: Proc. Int'l Workshop on Data Warehousing and OLAP, pp. 14–21 (2002)

21. Mesiti, M., Valtolina, S.: Towards a user-friendly loading system for the analysis of big data in the internet of things. In: IEEE Computer Software and Applications Conference - Workshops, pp. 312–317 (2014)
22. Ceri, S., Cochrane, R., Widom, J.: Practical applications of triggers and constraints: Success and lingering issues. In: Proc. of VLDB 2000, pp. 254–262 (2000)
23. Behrend, A., Dorau, C., Manthey, R.: SQL triggers reacting on time events: an extension proposal. In: Grundspenkis, J., Morzy, T., Vossen, G. (eds.) ADBIS 2009. LNCS, vol. 5739, pp. 179–193. Springer, Heidelberg (2009)
24. Bertino, E., Cochinwala, M., Mesiti, M.: UCS-Router: a policy engine for enforcing message routing rules in a universal communication system. In: Proc. of Mobile Data Management 2002, pp. 8–16 (2002)
25. Jin, Y., Bhavsar, T.: Incorporating fuzziness into timer-triggers for temporal event handling. In: Proc. of IRI 2008, pp. 325–329 (2008)